EXHIBIT T

POSTMORTEM OF SERVICE OUTAGE AT 3.4M CCU

2.8.2018

By The Epic Team

Fortnite hit a new peak of **3.4 million concurrent players** last Sunday… and that didn't come without issues! This blog post aims to share technical details about the challenges of rapidly scaling a game and its online services far beyond our wildest growth expectations.

Also, Epic Games needs YOU! If you have domain expertise to solve problems like these, and you'd like to contribute to Fortnite and other efforts, join Epic in **Seattle**, **North Carolina**, **Salt Lake City**, **San Francisco**, **UK**, **Stockholm**, **Seoul**, or elsewhere! Please shoot us an email at **OnlineJobs@epicgames.com**.

POSTMORTEM OF FEBRUARY 3RD AND 4TH OUTAGES

The extreme load caused 6 different incidents between Saturday and Sunday, with a mix of partial and total service disruptions to Fortnite.

MCP DATABASE LATENCY

Fortnite has a service called MCP (remember the Tron nemesis?) which players contact in order to retrieve game profiles, statistics, items, matchmaking info and more. It's backed by several sets of databases used to persistently store this data. The Fortnite game service is our largest database to date.

The primary MCP database is comprised of 9 MongoDB shards, where each shard has a writer, two read replicas, and a hidden replica for redundancy. At a high level user specific data is spread across 8 shards, whereas the remaining shard contains matchmaking sessions, shared service caches, and runtime configuration data.

The MCP is architected such that each service has a db connection pool to a sidecar process that in turn maintains a connection pool to all of our shards. At peak the MCP handles 124k client requests per second, which translates to 318k database reads and 132k database writes per second with a sub 10ms average database response time. Of that, matchmaking requests account for roughly 15% of all db queries and 11% of all writes across a single shard. In addition our current matchmaking implementation requires data to be in a single collection.

At peak we see an issue where the matchmaking shard begins queuing writes waiting on available writer resources. This can cause db update times to spike in the 40k+ ms range per operation causing MCP threads to block. Players experience unusually long wait times not just attempting to matchmake, but with all operations. We have investigated this in detail and it is currently unclear to us and support why our writes are being queued in this way but we are working towards a root cause.

This issue does not recover and the db process soon becomes unresponsive, at which point we need to perform a manual primary failover in order to restore functionality. During these outages this procedure was being repeated multiple times per hour. Each failover causing a brief window of matchmaking instability followed by recovery.

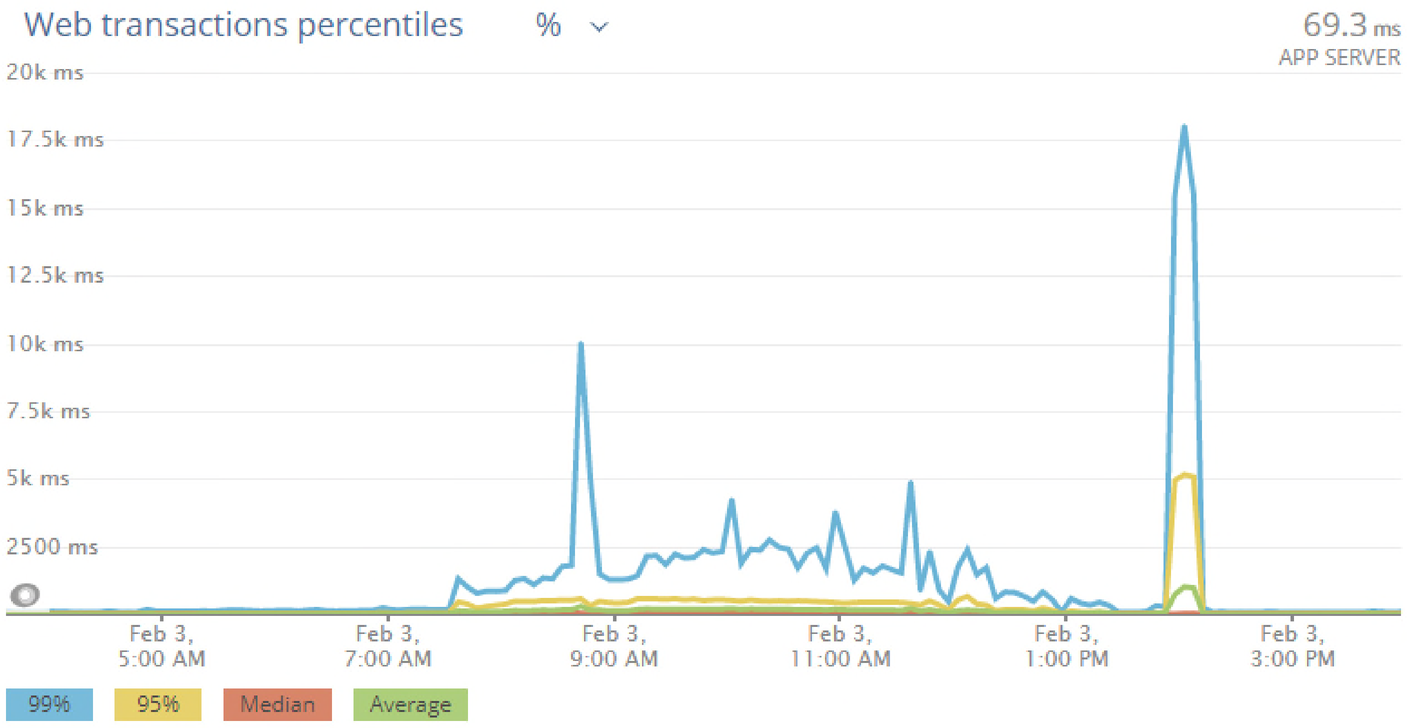
MCP THREAD CONFIGURATION

Prior to the launch of Fortnite, we had made a change to the packaging of the MCP. As part of that we introduced a bug limiting the number of available service threads below what we considered to be a safe default for our scale at the time. As part of a recent performance pass, this mistake was corrected by reverting it to our previous intended value.

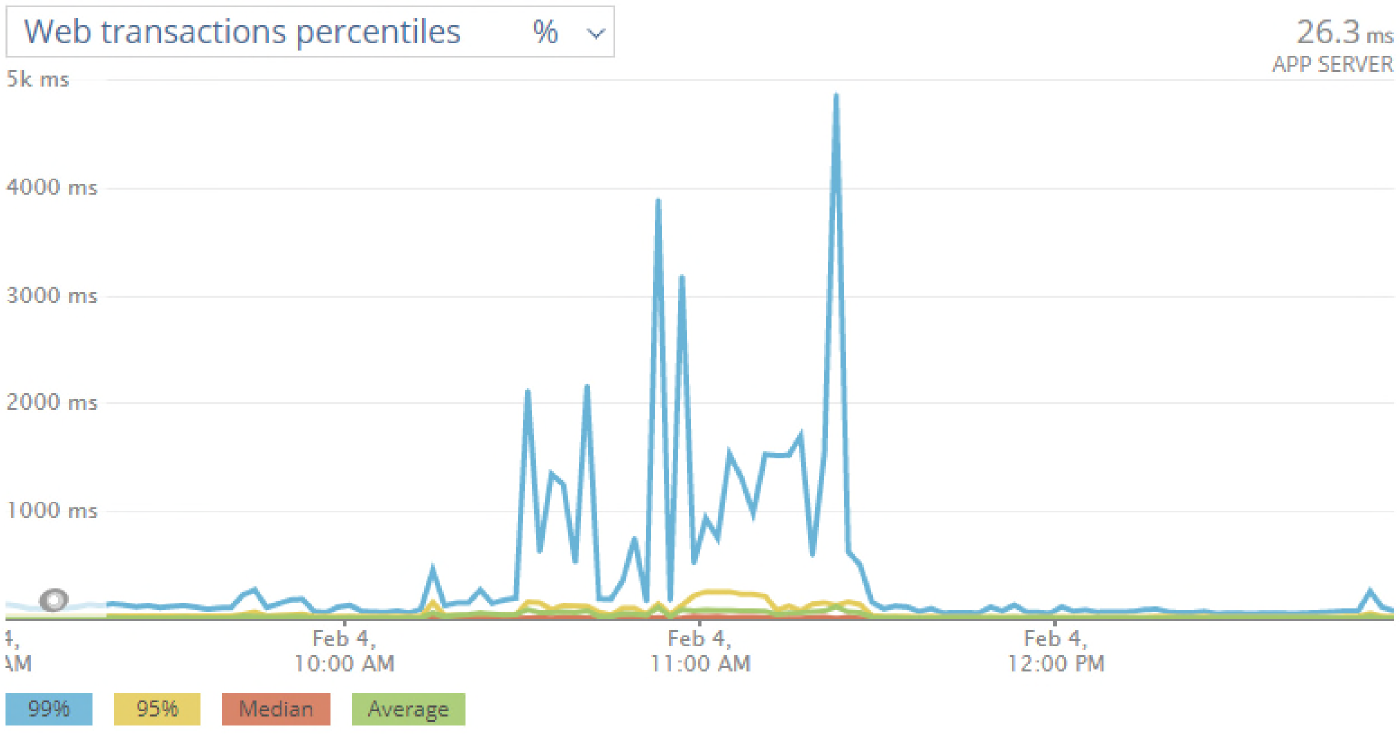
However once deployed to our live environment, we noticed requests experiencing increased latency (double ms average to double seconds) that was not present in our pre-production environments. This was diagnosed as db connection pool starvation via real-time cpu sampling through a diagnostics endpoint. In order to quickly remediate the issue we rolled back to our previous thread pool configuration.

What we expected to be a performance improvement resulted in the opposite and was only revealed at peak production workloads.

The above MCP issues on Saturday can be seen here, with spikes partially representing matchmaking db failures and overall poor performance due to db thread pool starvation and a gradual rolling deploy.



The impact of just matchmaking db failures on Sunday can be seen below:

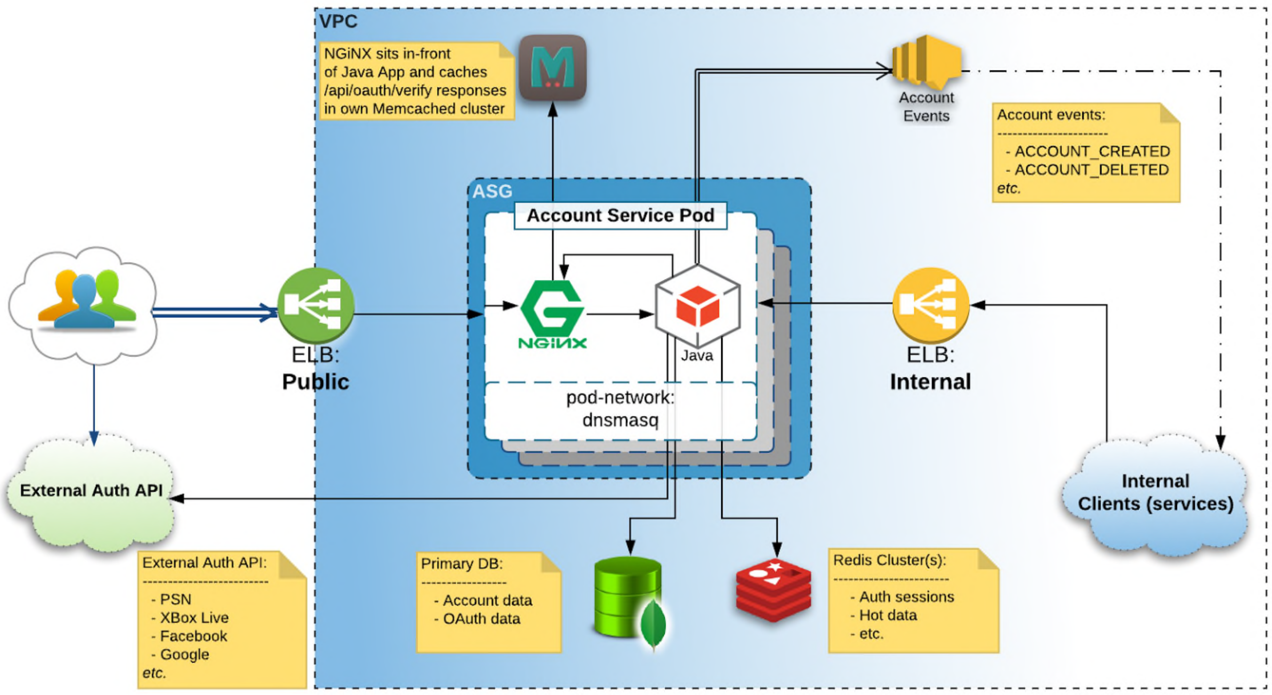


ACCOUNT SERVICE OUTAGE

Account Service is the core Epic service which maintains user account data and serves as an authentication endpoint. Service in numbers:

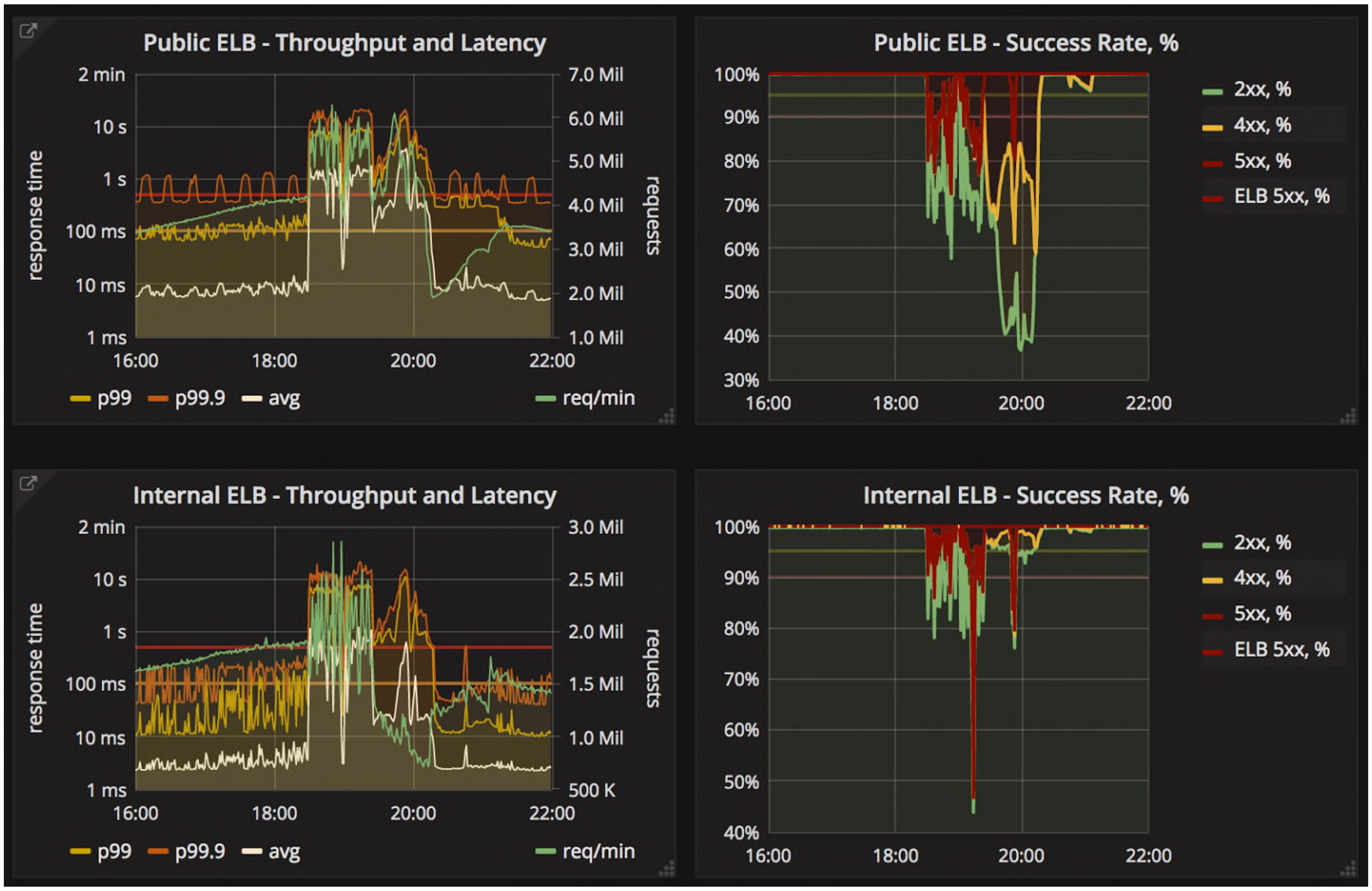
- Throughput: 40k - 100k req/sec (2.5M - 6M rpm). Up to 160k/sec at land rush.
- Latency:
    - All APIs: avg < 10ms, p99 < 100ms, p99.9 < 400ms
    - Sign In: avg < 100ms
    - Auth check: avg < 5ms
- Sign up: 1k - 3k a minute.
- Sign in/out: 0.5k - 1k fresh sign ins /sec . Plus 0.7k - 1.5k refreshes /sec (session extension).

Account Service is a complex application with a JAXRS-based web-service component and a number of sidecar processes, one of which is an Nginx proxy sitting in front of it.



The main purpose of this proxy is to shortcut an access token verification path. All traffic is routed through this proxy, but only access token verification traffic is checked against cache as shown above. With all other calls simply passing through to the main application.

On Sunday, there was an incident when Memcached instability saturated Nginx capacity (essentially, occupied all available worker threads), so that other traffic simply couldn't get through to the main application.



Below is a quick post-mortem summary:

- Timeline:
  - 2018-02-04 18:30 UTC - 2018-02-04 20:20 UTC
- Root cause:
  - Nginx in front of the Java application was saturated, and traffic was limited to the application. Hence all our JVM-level protection from situations like this didn't help.
- Incident Details:
  - Our memcached component started failing under the load due to network and connection saturation.
  - Nginx was next in line and got stuck on timing out Memcached calls (100ms) quickly running out of free worker threads and becoming unable to serve other traffic. Including health-check calls.
  - All the verify calls have fallen through to JVM layer with added +100ms memcached timeouts and added to Redis load.
  - Missed health check requests caused the load balancer to pull all the nodes out of rotation effectively imposing full service downtime.

- Impact:
    - Good news though is that this had moderate impact on players in match due to resiliency measures we have in place.
    - Signing in (and out) was mostly blocked and our Epic Games Launcher would sign players out with an error.
- Next steps:
    - Leveraging Level 7 routing in ALB to direct all non-verify traffic directly to our Java application, which in turn has a number of protection measures implemented against saturation like this.
    - Significantly increasing Memcached capacity.
    - Followed by removing Nginx + Memcached couple altogether out of equation.
    - We still have a lot of exciting problems to solve. Problems like implementing effective sharding of persistent data with pretty complex set of secondary indices we have to maintain.

XMPP OUTAGE

Being a foundation for online presence, text messaging and a number of other social features like parties, XMPP Service plays a significant role in delivering quality social experience to our players. This makes all XMPP service instabilities immediately visible to our community.

XMPP Service is an Instant Messaging solution customized to support a subset of XMPP protocol **(https://en.wikipedia.org/wiki/XMPP)** and protocol extensions according to platform needs.

XMPP Service in numbers:

- Online connections: way over 3 Million ++
- Throughput in packets:
    - Total: ~600k / sec (including aux traffic, forwarding and broadcast effect).
    - Presences: ~180k / sec
    - Notification: ~50k / sec
    - Messages: ~40k / sec

We leverage XMPP for the following features:

- Online presence
- Push notifications
- Whispers

- Group chat - for parties, for team chat and global rooms.

In its essence XMPP, as majority of other instant messaging services, is a highly async pub-sub system pumping packets - messages, presences, commands and various aux data - through the cluster from a sender to an addressee (or a set of).

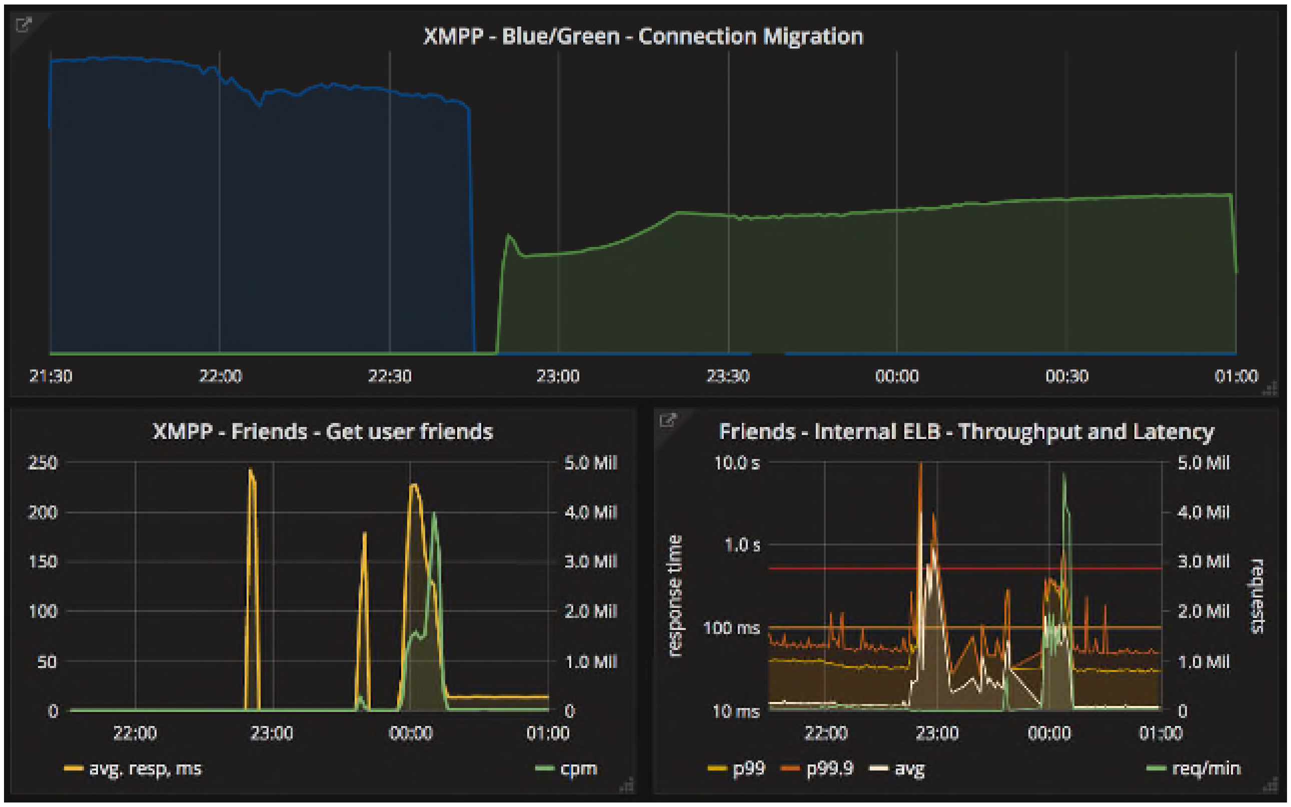
XMPP supports multiple end-user connection protocols. Our service uses two: TCP and WebSockets. We maintain millions of persistent and relatively long living TCP connections from clients concurrently. This comes at a cost of system complexity, as this case significantly differs from our typical RESTful web services.

Epic XMPP is one component in a family of Social web-services. It depends on other services including **Friends Service**, which supplies XMPP with friends information. We use this information to enable presence flow between players.

On Sunday we had a situation while mitigating a known instability problem that resulted in overloading a downstream system component and effectively paralyzing presence flow. Without presence, a user who is your friend cannot see that you are online, breaking most of our social features including the ability to form parties.

Graphs below:

- At the top - instability and connection migration from Blue cluster to Green cluster (time in UTC).
- On the left - calls made by stand-by XMPP cluster to Friends Service. XMPP perspective.
- On the right - traffic handled by Friends load balancer. Once recovered, it was able to handle all queued traffic.



Here is a quick summary of the incident:

- Timeline:
  - 2018-02-04 22:00 UTC - 2018-02-05 00:15 UTC
- Root cause:
  - Friends Service internal load balancer - the one on critical path for XMPP - got overloaded and pushed into an error state.
  - ELB could not quickly recover due to specifics of failover process and outdated network configuration - ELB subnet was short on free IPs to provision replacement.
- Incident Details:
  - Due to a recently introduced memory leak, XMPP was on a monitored path to falling into unstable state.
  - We planned to replace it with another ready and stand-by cluster with the leak already fixed.
  - We expected cluster to survive through weekend, so that we could schedule a proper maintenance during working week days.
  - Unfortunately, Game Services and Account Service instability significantly increased the effect of the leak. And at 22:00 UTC on Sunday we started losing cluster nodes and disconnecting players.

  - A decision was made to immediately failover to a stand-by cluster via green/blue deployment strategy, when we instantly flip all the traffic to another set of endpoints.
  - Unfortunately, landrush of reconnecting people at the time has effectively killed one of the **Friends Service** load balancers paralyzing our ability to setup presence flow on new connections.
- Impact:
  - As a result, though people did actually connect to XMPP, the UI showed everyone as offline due to missing presence flow.
  - Effectively, a "dark room" situation.
- Next steps:
  - We're in a process of upgrading load-balancer solution for Friends Service and other platform services to address issues like above.
  - We're fixing our VPC configuration to ensure subnet capacity.
  - There are also longer-term problems team is actively working on. For example with current architecture XMPP cluster represents a full mesh. Each cluster node is connected to each other. With 10 connections between each node and 101 nodes in cluster it effectively spends 1k sockets per node just on cluster connections.
  - Each XMPP node can hold only up to N connections with current solution. Hence there is a theoretical limit on optimal number of cluster nodes (and hence CCU capacity) we can maintain without solution redesign.

CLOUD CAPACITY LIMITS AND THROTTLING

We run Fortnite's dedicated game servers primarily on thousands of c4.8xlarge AWS instances, which scale up and down with our daily peak of players. This means our count of instances is always fluctuating and nonlinear in growth.

While capacity limits caused no disruption to the game, we had to react quickly to adjust some of our services limits. Fortunately our monitoring alerted us quickly and we were able to make the necessary changes. The limit we hit was the total instance limit in the region, which would affect our ability to scale our services in the entire region. We have also hit several API rate limits and we cover our corrective actions in the next steps section.

AVAILABLE IP EXHAUSTION

We run in multiple availability zones in cloud providers for our core services and our standard subnets are /24 giving us 251 usable IPs per subnet. Multiple factors such as shared subnets, instance changes and scaling across many services caused us to run out of IPs. While we were able to shift many components without any interruption, due to other events described above, it caused extended load balancer recovery times.

NEXT STEPS AND UPDATES

Our top focus right now is to ensure service availability. Our next steps are below:

- **Identify and resolve the root cause of our DB performance issues.** We've flown Mongo experts on-site to analyze our DB and usage, as well as provide real-time support during heavy load on weekends.
- **Optimize, reduce, and eliminate all unnecessary calls to the backend from the client or servers.** Some examples are periodically verifying user entitlements when this is already happening implicitly with each game service call. Registering and unregistering individual players on a game play session when these calls can be done more efficiently in bulk, Deferring XMPP connections to avoid thrashing during login/logout scenarios. Social features recovering quickly from ELB or other connectivity issues. When 3.4 million clients are connected at the same time these inefficiencies add up quickly.
- **Optimize how we store the matchmaking session data in our DB.** Even without a root cause for the current write queue issue we can improve performance by changing how we store this ephemeral data. We're prototyping in-memory database solutions that may be more suited to this use case, and looking at how we can restructure our current data in order to make it properly shardable.
- **Improve our internal operation excellence focus in our production and development process.** This includes building new tools to compare API call patterns between builds, setting up focused weekly reviews of performance, expanding our monitoring and alerting systems, and continually improving our post-mortem processes.
- **Improve our alerting and monitoring of known cloud provider limits, and subnet IP utilization.**
- **Reducing blast radius during incidents.** A number of our core services are globally impacting to all players. While we operate game servers all over the world, expanding to additional cloud providers and supporting core services in multiple geographical locations will help reduce player impact when services fail. Expanding our footprint also increases our

operational overhead and complexity. If you have experience in running large worldwide multi cloud services and/or infrastructure we would love to hear from you.

- **Rearchitecting our core messaging stack.** Our stack wasn't architected to handle this scale and we need to look at larger changes in our architecture to support our growth.
- **Digging deeper into our data and DB storage.** We hit new and interesting limits as our services grow and our data sets and usage patterns grow larger and larger every day. We're looking for experienced DBAs to join our team and help us solve some of the scaling bottlenecks we run into as our games grow.
- **Scaling our internal infrastructure.** When our game services grow in size so do our internal monitoring, metrics, and logging along with other internal needs. As our footprint expands our needs for more advanced deployment, configuration tooling and infrastructure also increases. If you have experience scaling and improving internal systems and are interested in what is going on here at Epic, let's have a chat.
- **Performance at scale.** Along with a number of things mentioned, even small performance changes over N nodes collectively make large impacts for our services and player experience. If you have experience with large scale performance tuning and want to come make improvements that directly impact players please reach out to us.
- **MCP Re-architecture**
    - Move specific functionality out of MCP to microservices
    - Event sourcing data models for user data
    - Actor based modeling of user sessions

Problems that affect service availability are our primary focus above all else right now. We want you all to know we take these outages very seriously, conducting in-depth post-mortems on each incident to identify the root cause and decide on the best plan of action. The online team has been working diligently over the past month to keep up with the demand created by the rapid week-over-week growth of our user base.

While we cannot promise there won't be future outages as our services reach new peaks, we hope to live by this great quote from Futurama, "**When you do things right, people won't be sure you've done anything at all.**"

REMEMBER, WE NEED YOU!

It's been an amazing and exhilarating experience to grow Fortnite from our previous peak of

60K concurrent players to 3.4M in just a few months, making it perhaps the **biggest PC/console game in the world**! All of this has been accomplished in just a few months by a small team of veteran online developers -- and we'd love to welcome a few more folks like yourself to join Epic Games on this journey!

So, please contact us at OnlineJobs@epicgames.com, and join one of Epic's high-quality development offices around the world!

❮ PREVIOUS ARTICLE (/fortnite/en-US/blog/v2-4-2-patch-notes) (/fortnite/en-US/blog)

NEXT ARTICLE (/fortnite/en-US/blog/fortnite-battle-royale-state-of-development-v4) ❯

(https://www.facebook.com/FortniteGame) (https://twitter.com/FortniteGame)

(https://www.twitch.tv/fortnitegame) (https://www.youtube.com/epicfortnite)

(https://www.instagram.com/fortnite/) (https://vk.com/fortnite)

Home (/fortnite/en-US/home)
Battle Pass (/fortnite/en-US/battle-pass)
Watch (/fortnite/en-US/watch-fortnite)
Get Fortnite (/fortnite/en-US/buy-now/battle-royale)

News (/fortnite/en-US/news)
FAQ (/fortnite/en-US/faq)
EULA (/fortnite/en-US/eula)
Competitive (/fortnite/competitive/en-US/home)

V-Bucks Card (/fortnite/en-US/vbuckscard)
Help (/fortnite/en-US/help-center)
Safety and Security (/fortnite/en-US/safety-and-security)
Community Rules (https://www.epicgames.com/site/community-rules)

US/CANADA

TEEN ESRB Violence
In-Game Purchases / Users Interact

Terms of Service (https://www.epicgames.com/en-US/tos?lang=en-US) Privacy Policy (https://www.epicgames.com/en-US/privacypolicy?lang=en-US)